

ABBUC HARDWARE COMPETITION 2014

Marcin Sochacki (Montezuma)

SIO2BT Manual v2.0

Many Thanks to:

Bernd, Bob!k, Dietrich, drac030, FlashJazzCat, HardwareDoc, Hias, Kr0tki, Lotharek, xxl

Table Of Contents

Introduction.....	3
Wiring Diagrams.....	5
External wiring.....	5
Internal Wiring.....	6
List of Materials.....	8
Modifications in Atari OS.....	8
Flashing Ultimate 1MB with BT-enabled OS.....	9
Patching Atari OS at runtime.....	14
Bluetooth Module configuration.....	15
Introduction to Bluetooth.....	15
Bluetooth under Windows.....	16
Bluetooth under Linux.....	18
Emulating SIO devices with SIO2BSD.....	21
Emulating SIO devices with AspeQt.....	21
Emulating SIO devices with SIO2BT Android App.....	23
Sparta DOS X support for SIO2BT.....	28
References.....	28

Introduction

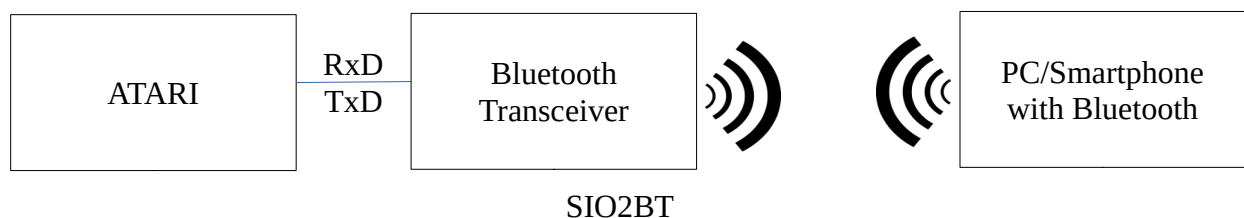
SIO2BT project is a set of hardware and software solutions related to the wireless Bluetooth communication between the 8-bit Atari computers and Bluetooth (BT) enabled Serial Input Output (SIO) devices.

Due to delay in the Bluetooth data transmission it was necessary to modify the original Atari OS. For that reason the timeout values (which could not be met) were increased. A tool for patching Atari OS rom files (for Atari 800 and Atari XL/XE) is provided as a part of the project. Additionally there are disk images available for Atari XL/XE for those who want to use SIO2BT with original, not modified Atari computers.



Another difficulty was missing information about the SIO Command Line status. Hardware handshaking had to be replaced with a software solution for Command Frame detection.

Most of the Bluetooth transceivers available on the market are based on the CSR BC417143 chipset supporting SPP (Bluetooth Serial Port Profile). Two Bluetooth transceivers plugged between any two legacy serial devices would (in theory) be transparent and would appear as if they were connected via cable. In practice missing hardware handshaking and non-deterministic delays cause problems if the protocols rely on timing or on handshaking. And this is exactly the case with the SIO Protocol.



The BT transceivers provide RxD and TxD lines (TTL voltage level), which can be connected directly to the DATA OUT and DATA IN lines of the Atari SIO port.

Once configured for the SIO Communication (Baudrate = 19200, etc.), a Bluetooth transceiver can be paired with a PC. When a device is paired, the Bluetooth Software Stack on the PC creates a virtual serial port for it. Opening such ports establishes a Bluetooth connection between the PC and the transceiver. A PC can run the software emulating SIO devices.

I have adapted one of the most popular tools for emulating SIO devices – AspeQt 0.8.8 – for communication via Bluetooth. The tool is licensed under GNU General Public License version 2.0 (GPLv2) and my source code modifications are available freely to everyone.

The modifications include:

- a bug fix for Windows to support COM port numbers higher than 9

- a removal of a background image in the logger window, which consumes a lot of the CPU power (especially visible with weak machines like Raspberry Pi)
- software command frame detection (SOFTWARE handshaking)
- configurable delay for writing data to the Atari

The additional delay for writing is required for Bluetooth, because an Atari expects (according to the SIO specification) a minimum time delay between the “Acknowledge” Byte and the “Complete” Byte.

This timing is critical and unfortunately can not be guaranteed by Bluetooth. If we send two bytes waiting X milliseconds in between, the first byte can get stuck somewhere in the Bluetooth stack and the second byte will catch it up so they will arrive one after each other at the Atari. The Atari could skip the second byte and keep waiting for it. Luckily (after a certain timeout) the last Command Frame would be repeated and the communication would continue.

Using a Logic Analyzer and „trial and error“ approach I had to face the fact that there is no 100% guarantee for fast and error free communication. I can recommend a delay value of 10 milliseconds (default setting) for the most cases and bigger values for machines with slower Bluetooth stacks.

The SOFTWARE handshaking can also be used (even without any delay) with cheap USB2Serial cables (TTL) that do not support hardware handshaking.

There is one drawback of the SOFTWARE handshaking that should be understood by the users. The emulated SIO device is analyzing data coming from the Atari to detect an incoming Command Frame. We are on the safe side if our device is the only SIO device on the bus. However if the Atari writes data to a different physical SIO devices on the bus, the data (for example a disk sector) can include bytes that build a valid Command Frame. In such cases our device would incorrectly react upon it.

My recommendation is NOT TO USE other physical SIO devices when working with SOFTWARE handshaking, or to use them only to provide data to Atari.

As a part of the project I have provided a binary version of the AspeQt for Windows users and the source code for Linux users (tested successfully under Windows XP, Windows 7 and Ubuntu).

As long as the SIO devices are emulated on a PC, there is no real advantage of wireless communication.

However if SIO devices are emulated on a smartphone, we do not need a PC at all. Many people carry smartphones with them. Games can be downloaded from the internet and immediately loaded to the Atari from an emulated Disk Drive. One day we could even go one step further and introduce a new SIO Device – an intelligent network device with a TCP/IP stack...

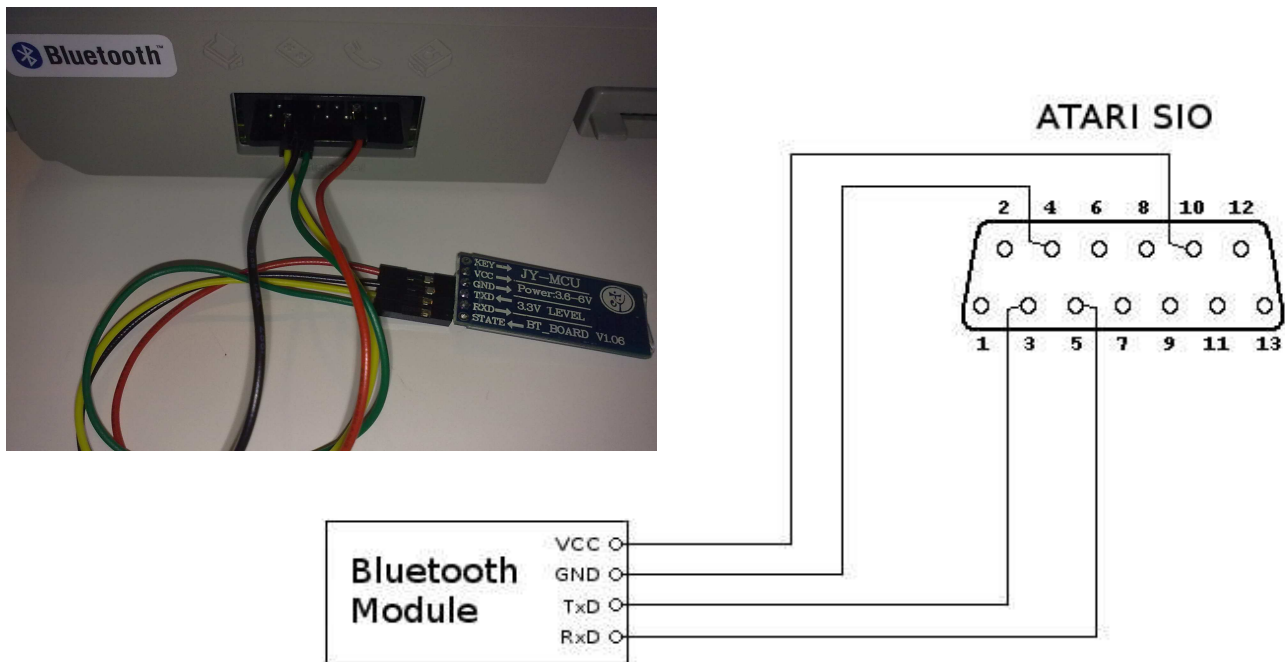
I decided to support the Android platform (the iOS does not support Bluetooth SPP).

The SIO2BT App can be downloaded from the Google Play Store to any smartphone with Android 2.1 (API 7) or higher.

Wiring Diagrams

External wiring

If you decide to use SIO2BT like an external Bluetooth dongle, the wiring is pretty straightforward:



(looking at the SIO connector from behind the computer)

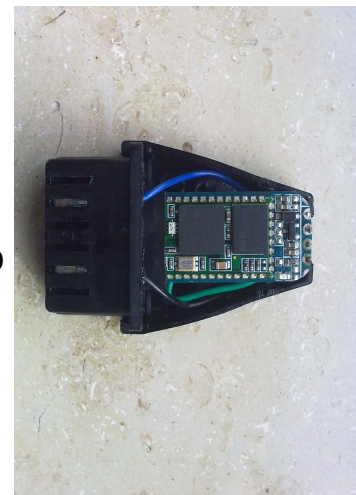
The obvious advantage of external wiring is simplicity. The Bluetooth Module is so small that it fits into a SIO Plug.

The hardware is powered from the SIO connector. Every time you switch off the Atari, the Bluetooth Module will be switched off as well and the Bluetooth connection will get lost. It is a disadvantage of this approach, since you have to re-connect after powering on your Atari.

You may think about the Bluetooth connection as a SIO cable. If a SIO cable is not plugged in, an Atari can not communicate with a Disk Drive. The same happens if there is no active Bluetooth connection in case of SIO2BT.

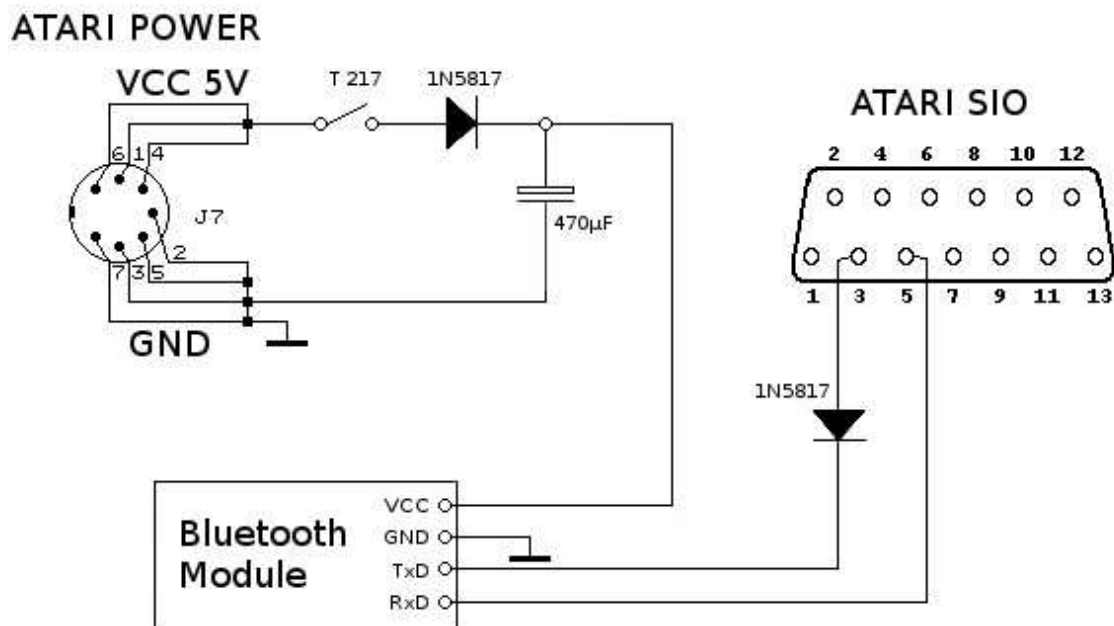
Note 1: There is intentionally no diode in the Data In line, because the SIO2BT should not be used in a SIO daisy chain with other SIO devices.

Note 2: Do not use Atari with SIO2BT without active Bluetooth connection (blinking LED), since data sent from the Atari could accidentally contain valid AT commands and could re-configure the module. See more details in the chapter about Bluetooth Module configuration.



Internal Wiring

With basic soldering skills, you will be able to install SIO2BT hardware inside your Atari:



The power is grabbed before the main Atari power switch and SIO2BT has its own power switch.

As soon as you connect the power supply unit to the Atari, your Bluetooth Module can be powered on and you can establish a wireless connection.

Switching off the Atari does not drop the Bluetooth connection.

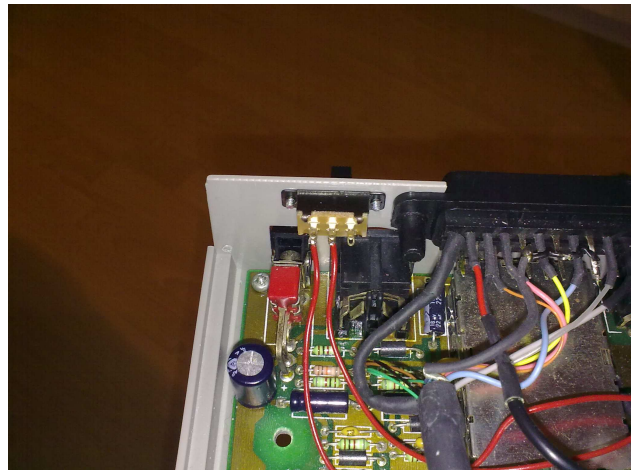
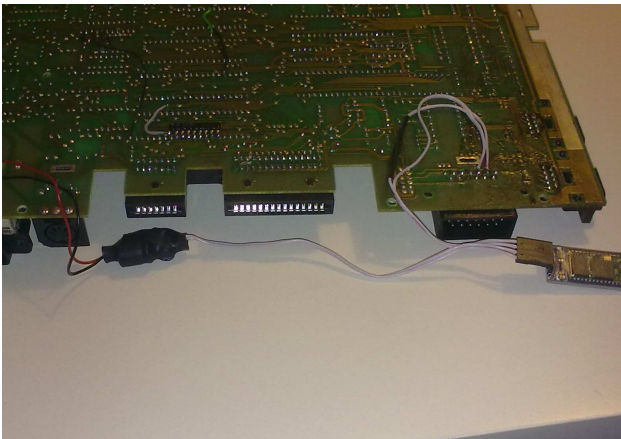
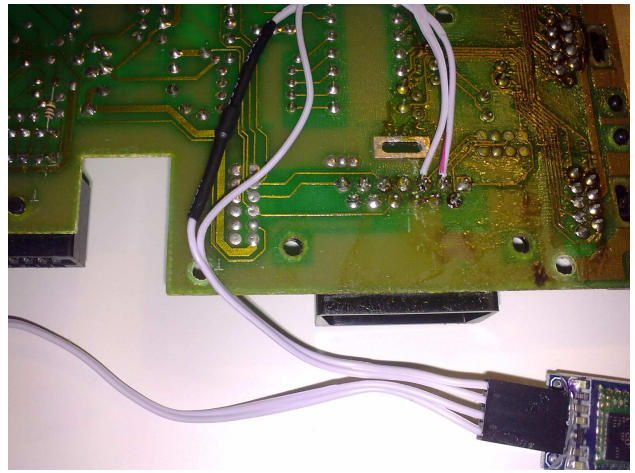
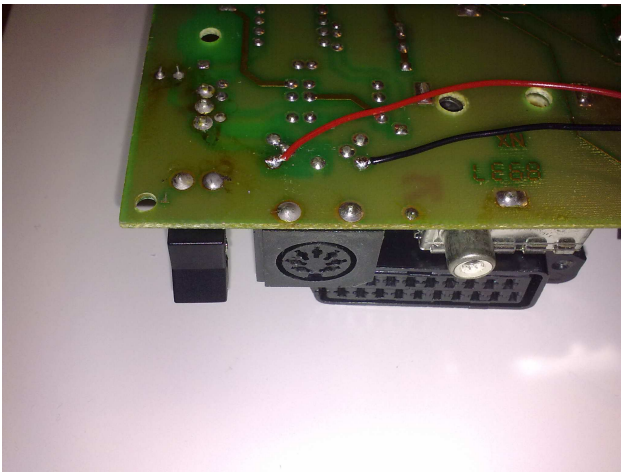
The diode and the capacitor stabilize voltage for the BT Module.

If you do not want to work with SIO2BT you can just switch it off with SIO2BT power switch.

The diode in Data In line makes possible to use SIO2BT with other SIO devices in a daisy chain. This is actually not recommended, however reading data from other SIO devices does not harm SIO2BT.

Note 1: Switch off SIO2BT when Atari is writing to other SIO devices.

Note 2: Do not use Atari with SIO2BT without active Bluetooth connection (blinking LED), since data sent from the Atari could accidentally contain valid AT commands and could re-configure the module. See more details in the chapter about Bluetooth Module configuration.



List of Materials

1) Bluetooth Transceiver, for example from Guangzhou HC Information Technology Co., Ltd.
(<http://www.wavesen.com/>)

2) Power switch, for example:

T 217 Schiebeschalter-Miniatur, Lötanschluß, 2x UM sw (<http://www.reichelt.de/>)

3) Capacitor 470 µF, for example:

RAD 470/16 Elektrolytkondensator, 10x12,5mm, RM 5,0mm (<http://www.reichelt.de/>)

4) 2 x Schottky diode, for example:

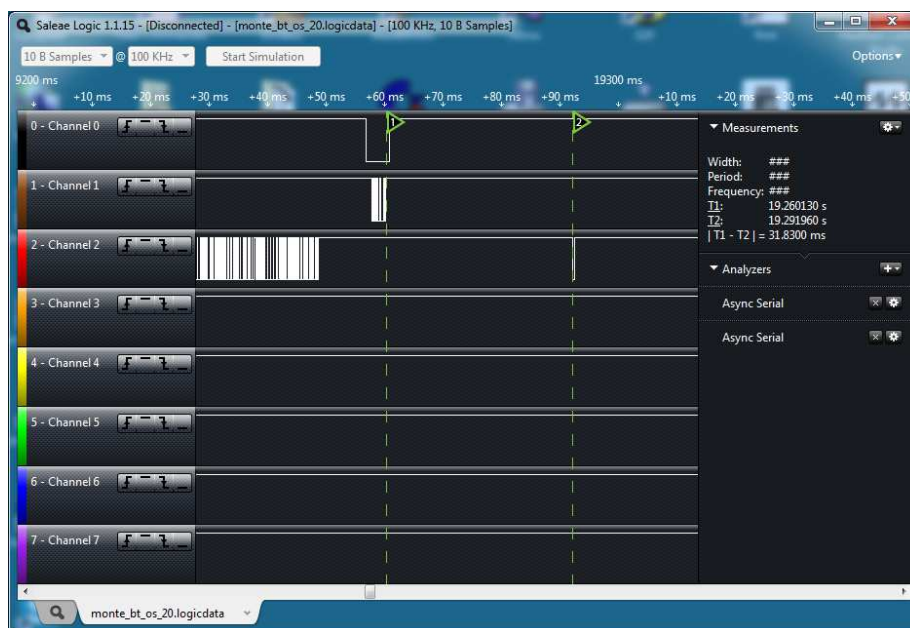
1N 5817 Schottky Diode, DO41, 20V, 1A (<http://www.reichelt.de/>)

5) PC with a Bluetooth dongle (Windows/Linux) or an Android smartphone

Modifications in Atari OS

The SIO Protocol defines the maximum time that Atari should wait for an acknowledge of a command frame. It is 16 milliseconds (for NTSC variant).

Due to delay in the Bluetooth transmission it was necessary to modify these values in the Atari OS.



The timeout was increased to 240 milliseconds (the biggest delay ever observed with a logic analyzer was about 150 milliseconds). This change influences dramatically the start-up time if no floppy is connected to the Atari. There are two loops in the OS code handling SIO communication. The outer loop handles a complete SIO sequence (command frame / data frame) and is repeated one time per device in error case. The inner loop handles sending a command frame and is repeated up to 13 times in error case (missing Acknowledge Byte). This makes 28 command frames when no disk drive is connected.

There is an extension of the SIO protocol introduced in the XL/XE OS for handling relocatable

peripheral handlers. So called “Type 3 Poll Command” is issued at power-on or reset time. This means additional 56 command frames. The only device supporting this new feature is Atari 1090, which exists only as a limited number of prototypes.

To preserve the original start-up time and to increase the timeout values at the same time, the new polling extension was disabled and the number of retries in the inner loop was reduced to 5. This makes only 12 command frames in total.

With a PC tool **SIO2BT_OS_Patcher.jar** you can patch Atari OS ROM files (for Atari 800 and Atari XL/XE) and the QMEG 4.04 ROM file. You just need to download the original ROM file and patch it with SIO2BT_OS_Patcher.



The selected ROM file will be cloned and the patched version will be saved under a new name. Example: XL.ROM > XL.ROM_PATCHED

The Atari OS ROMs can be downloaded from:

<http://www.atari Preservation.org/websites/freddy.offenga/osromv36.zip> (all Atari ROM files)

<http://sourceforge.net/projects/atari800/files/ROM/> (Atari XL ROM file)

Bios4Config owners can use Megacart Flash 512K modules to program the flash memory with the updated content (you may use **Bios4Config.jar** utility to create a 512K ROM file for flashing).

The owners of very popular **Ultimate 1MB** or **Incognito** extension boards can upload the patched OS to the board's flash memory with the **uFlash** utility written by FlashJazzCat:

<http://www.atari8.co.uk/uflash/index.html>

Flashing Ultimate 1MB with BT-enabled OS

The following screenshots show the procedure step by step.

Ultimate Setup

```

Extended memory: 1088k RAMBO
System: Stock OS
SpartaDOS X: Enabled
UBXE base: 0xD640
Stereo sound: Disabled
COVOX: Disabled
SIDE hardware: Disabled
SIDE PBI ID: 1
BASIC slot is: BASIC
XEGS GAME slot: Missile Command
Thr, 02-10-2014, 21:12:24
[←] move cursor, [→] change option
[S] to save, [Q] to quit
  
```

Ultimate clock installed

```

SpartaDOS X 4.46 11-04-2014
Copyright (C) 2014 by FTE & DLT

D1:DIR
Volume:
Directory: MAIN

UFLASH XEX 19405 2-10-14 20:33
XLBT ROM 16384 2-10-14 20:33
65355 FREE SECTORS

D1:UFLASH.XEX
  
```

Device Slot Help

```

Device: Ultimate
Flash ROM: A29040
Capacity: 512 KB
Sector: 64 KB
  
```

OK

[Tab] to move, [Ret] for OK, or [Esc]

Device Slot Help

```

Entire ROM
SpartaDOS X
GUI
BIOS
SIDE Loader
PBI BIOS
XL/XE 05 1: KMK 65816 05
XL/XE 05 2: Diagnostics
XL/XE 05 3: Q-Meg 05
XL/XE 05 4: Stock OS
BASIC Slot 1: BASIC
BASIC Slot 2: CAR1
BASIC Slot 3: CAR2
BASIC Slot 4: CAR3
XEGS Slot 1: Missile Command
XEGS Slot 2: CAR1
XEGS Slot 3: CAR2
XEGS Slot 4: CAR3
  
```

18 Items. Press [Esc] for Menu

Go to the Ultimate menu
(HELP+RESET) and enable:

- StockOS
- extended memory
- SpartaDOSX.

Press S to save settings and Q to Quit

Prepare an ATR Image containing
uflash.xex and the patched OS ROM and
mount it as D1.

Using AspetQt you can also mount the
directory containing these files as a disk.

Start uflash.xex

uflash.xex will automatically detect your
board

Press „Return“

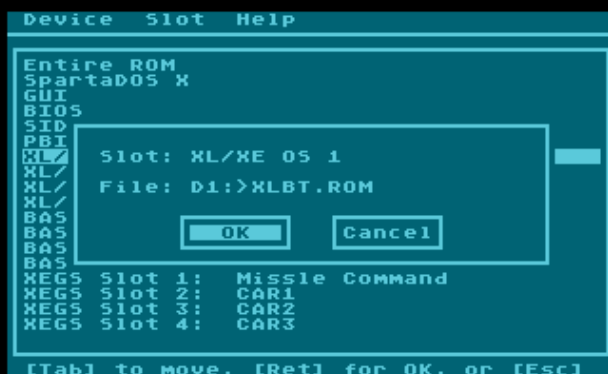
Select an OS slot, which you want to
overwrite.

Press „Return“



Use arrow keys to select the patched OS ROM file.

Press „Return“



Press „Return“ to confirm your selection



Reading progress is shown.



Flashing progress is shown.



Press „Return“



Use arrow keys to select the updated OS slot.

Press „Esc“ to show the uflash menu.

Navigate with arrow keys to the „Slot“ menu and select „Edit Name“ item.

Press „Return“



Edit the name and press „Return“

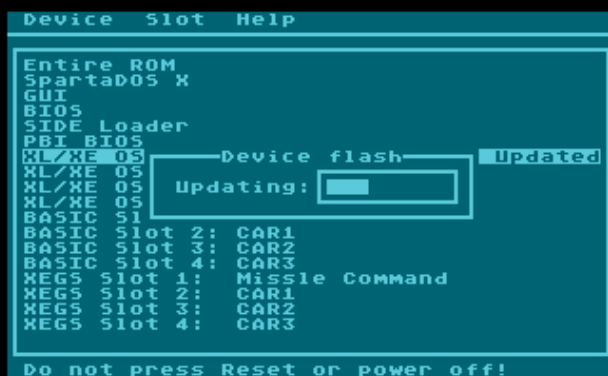


Press „Esc“ to show the uflash menu.

Navigate with arrow keys to the „Device“ menu and select „Flash Descriptions“ item.



Press „Return“ to confirm



Flashing progress is shown.



Press „Return“



Press „Esc“ to show the uflash menu.
Navigate with arrow keys to the „Device“ menu and select „Exit“ item.



Press „Return“ to confirm restart.

Go to the Ultimate menu
(HELP+RESET) and:
- disable SpartaDOSX.
- select XL-BT OS

Press S to save settings and Q to quit.
Your Atari is ready for SIO2BT :)

Patching Atari OS at runtime

Additionally there are disk images available for Atari XL/XE for those who want to use (with some limitations) SIO2BT with original, not modified Atari computers.

You can use the following images: 600XL.atr, 1200XL(a).atr, 1200XL(b).atr, XL_XE_XEGS.atr. Mount one of them (depending on your Atari) as D1 and a game to be loaded as D2.

Additionally there is one more set of disk images, which can be used to load ATR games: 600XL_SWAP.atr, 1200XL(a)_SWAP.atr, 1200XL(b)_SWAP.atr, XL_XE_XEGS_SWAP.atr. Mount one of them (depending on your Atari) as D1 and a game to be loaded as D2.

When the loader is executed, the screen color changes to dark blue. Now SWAP the disks on the Smartphone, so the game to be loaded is mounted as D1 and press any key on the ATARI to continue.

These small (144 bytes) bootable disk images contain only one sector with code for OS patching. Atari sends two SIO commands to an emulated floppy (D1) at start-up: \$53 Get Status and \$52 Read Sector 1. The most of the times the communication is successful (with standard SIO) and only sometimes Atari gets stuck for a while (and re-tries). When the requested sector is loaded, the OS SIO code is modified and the Atari continues to load from the second disk (D2).

However the best way to use SIO2BT with the original OS is the **XBIOS loader for bluetooth**.

The loader XBIOS_4_BT_700.atr (or XBIOS_4_BT_400.atr) can be mounted as D1 and the game(s) to be loaded as D2. The only difference between the two XBIOS versions is the address in memory, where the loader is located (\$700 or \$400).

Note1: Some games can not be loaded this way

Note2: The highest compatibility can be achieved with a patched OS (recommended way)

Bluetooth Module configuration

The default BT Module configuration was modified for the SIO2BT project.

As long as the Bluetooth connection is not established (LED is blinking) it is possible to re-configure the module with AT commands (over a serial cable):

```
AT+NAME SIO2BT 1050
```

```
AT+PIN1050
```

```
AT+BAUD5
```

The example commands above set the Bluetooth friendly name to “SIO2BT 1050”, set the PIN code to “1050” and the Baud rate to 19200.

Note: Do not use Atari with SIO2BT without active Bluetooth connection (blinking LED), since data sent from the Atari could accidentally contain valid AT commands and could re-configure the module.

Introduction to Bluetooth

The scope of the Bluetooth specifications is very wide. It covers hardware and software starting from the radio signal to the high level protocols for wireless, short-range communication.

Every Bluetooth enabled device has so called Bluetooth stack (realized partly in the hardware and partly in the software), which implements the Bluetooth specifications (usually only a subset of them). One of the most popular Bluetooth protocols is RFCOMM (Radio Frequency Communications). It was designed to emulate RS-232 serial ports, the SPP (Serial Port Profile) is based on it and it is available in virtually every Bluetooth stack (in particular in a Bluetooth Transceiver used in the SIO2BT project).

In order to establish a wireless connection between two Bluetooth devices, one of the devices has to enter discoverable mode and the other device has to perform a search (device discovery).

In our case the Bluetooth Transceiver automatically enters discoverable mode when powered on and we have to search for it.

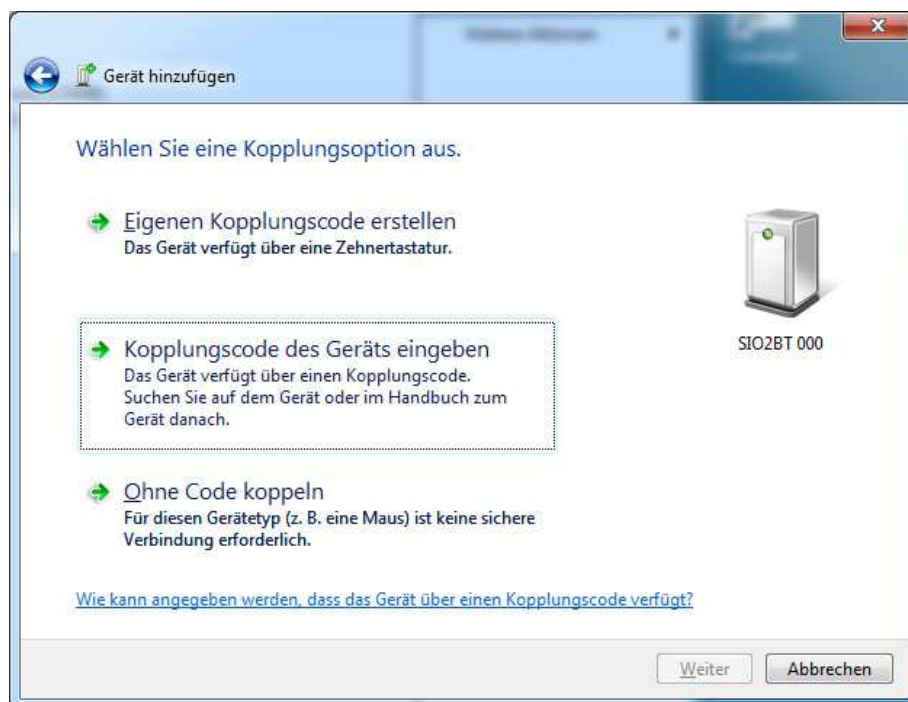
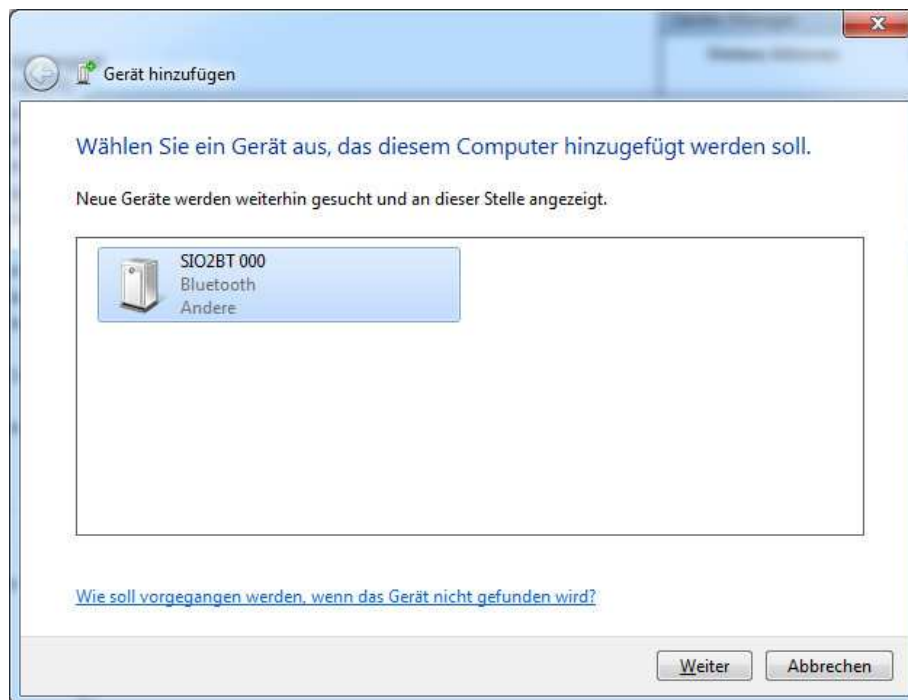
The first step in establishing the connection is an authentication. If devices do not know each other, they have to be paired first.

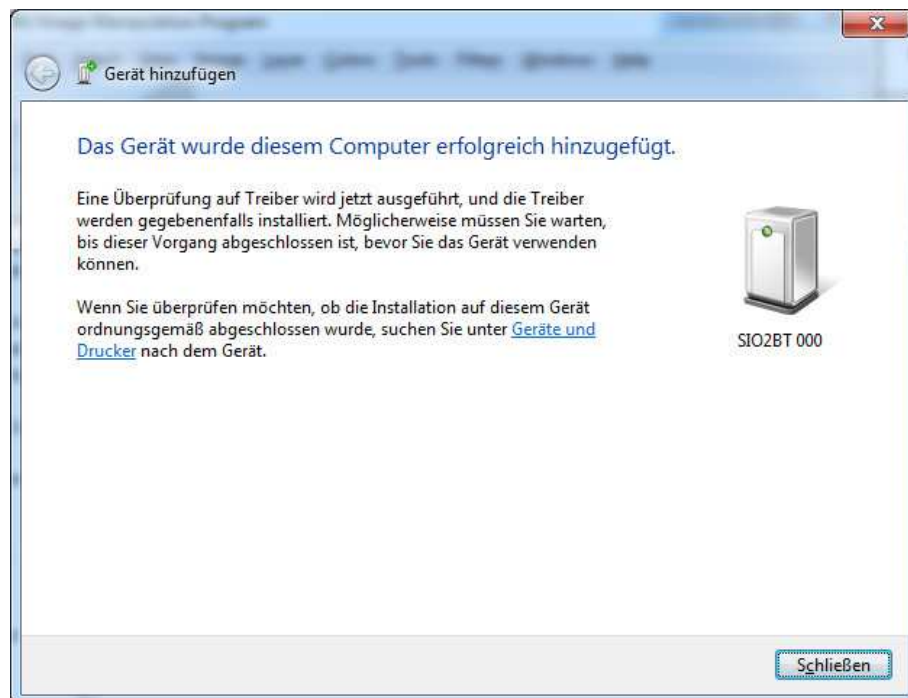
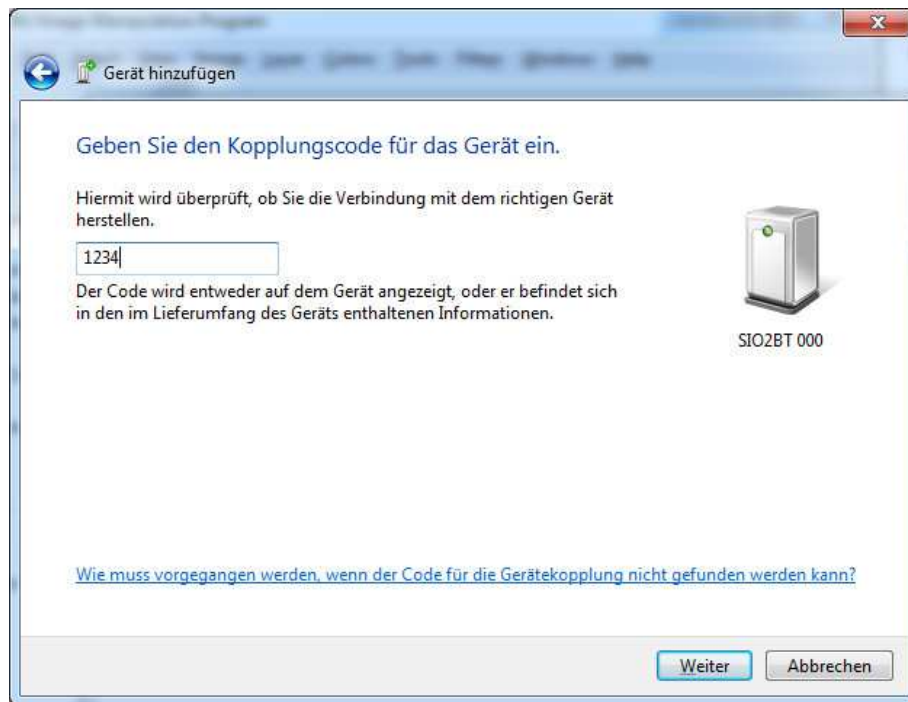
Pairing requires entering 4-digit PIN code of the Bluetooth Transceiver. After this number is entered both devices generate and store so called “Link Key” for future authentication.

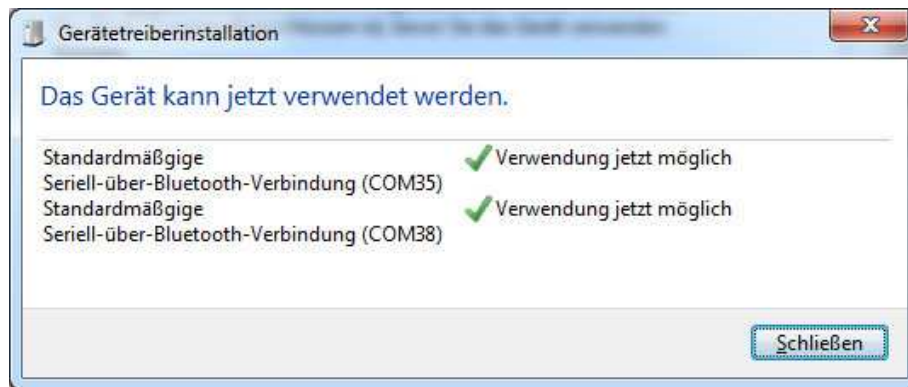
Once devices are paired, the wireless connection can be established. Under Windows and under Linux, the Bluetooth connection is established when a virtual serial port (created after pairing) is opened.

Bluetooth under Windows

Pairing process under Windows 7 is presented on the screen shots below:





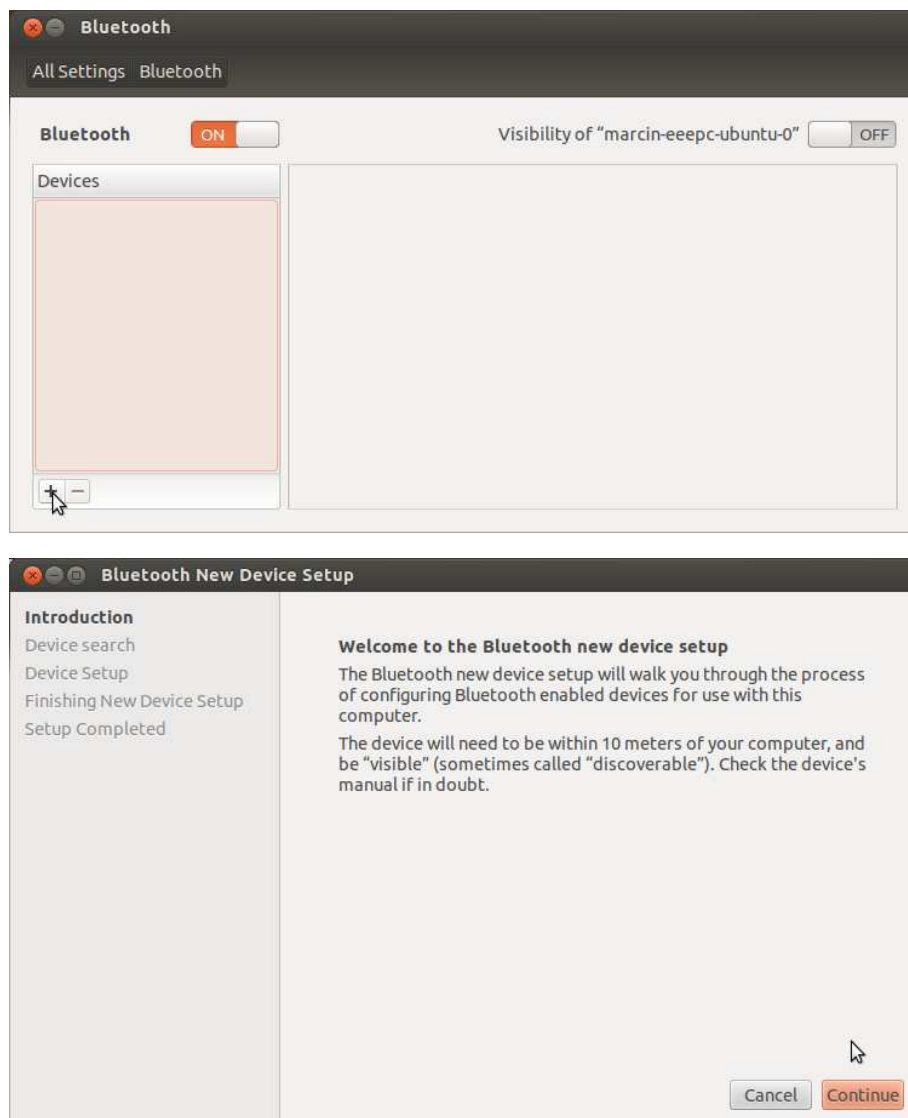


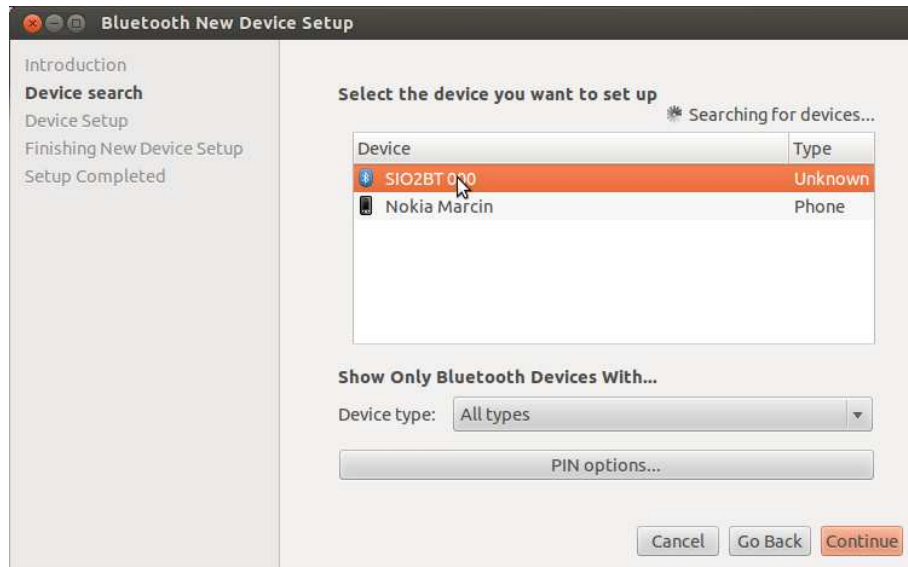
Windows automatically creates two virtual serial ports after successful pairing. Only the first of them is useful for SIO2BT.

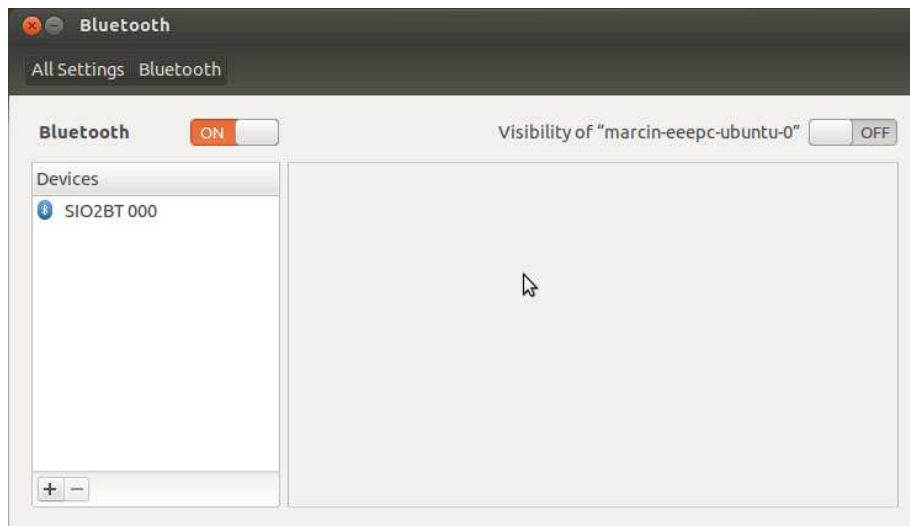
Please be patient, setting up virtual serial ports takes a few minutes!

Bluetooth under Linux

Pairing process under Ubuntu is presented on the screen shots below:







Ubuntu does not create virtual serial ports automatically after successful pairing.

Open a terminal and type in the following command:

```
$ sudo hcitool scan
```

The hcitool searches for discoverable devices and displays their Bluetooth addresses and Bluetooth friendly names:

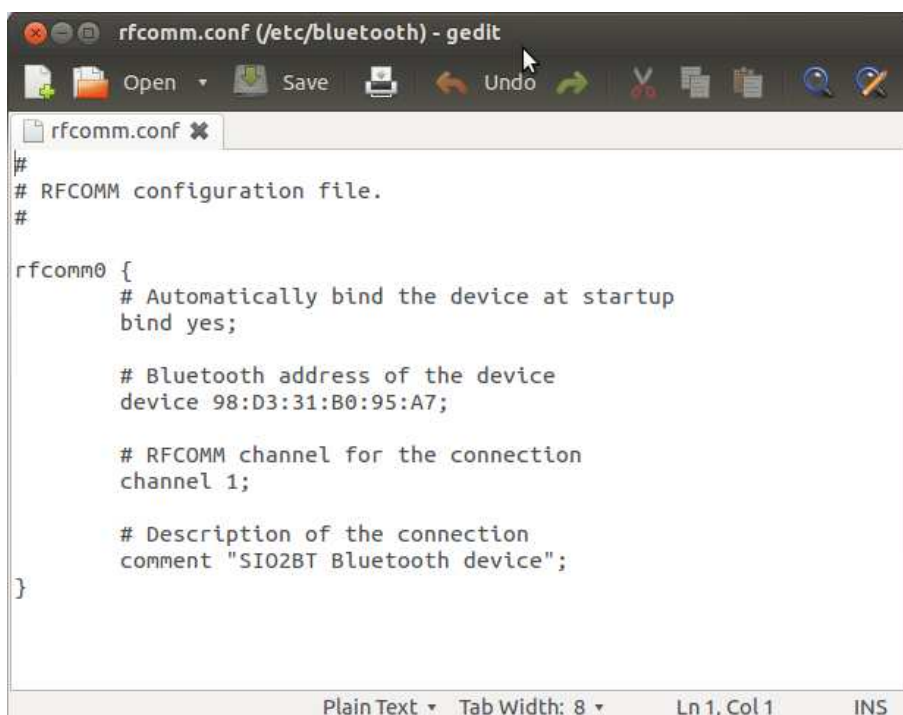
Scanning ...

```
98:D3:31:B0:95:A7 SIO2BT 000
```

You can create a virtual serial port for your SIO2BT by typing:

```
$ sudo rfcomm bind /dev/rfcomm0 98:D3:31:B0:95:A7
```

or you can let it happen automatically, by editing /etc/bluetooth file:



Note: Every Bluetooth device has a unique address, so you need to replace the example address above with the actual one of your SIO2BT Transceiver.

Your SIO2BT is accessible as `/dev/rfcomm0`

Now add the current user to the `dialout` group to give him access to the `rfcomm0` device:

```
sudo adduser $USER dialout
```

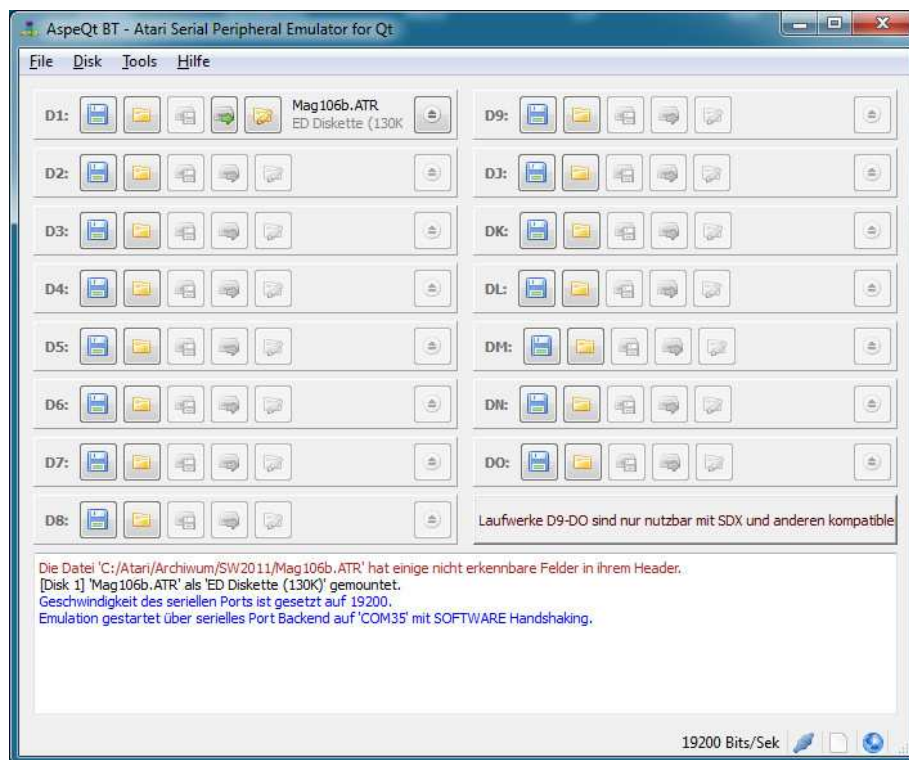
Emulating SIO devices with SIO2BSD

SIO2BSD is a PC tool that handles SIO communication under FreeBSD, Linux and MacOS X.

Example use: `sio2bsd -d10 -s /dev/rfcomm0 image.atr`

Emulating SIO devices with AspeQt

The Windows binary is provided with the SIO2BT project.



Linux users have to compile AspeQt first:

```
$ sudo apt-get install build-essential
```

```
$ sudo apt-get install qt4-dev-tools libqt4-dev libqt4-core libqt4-gui
```

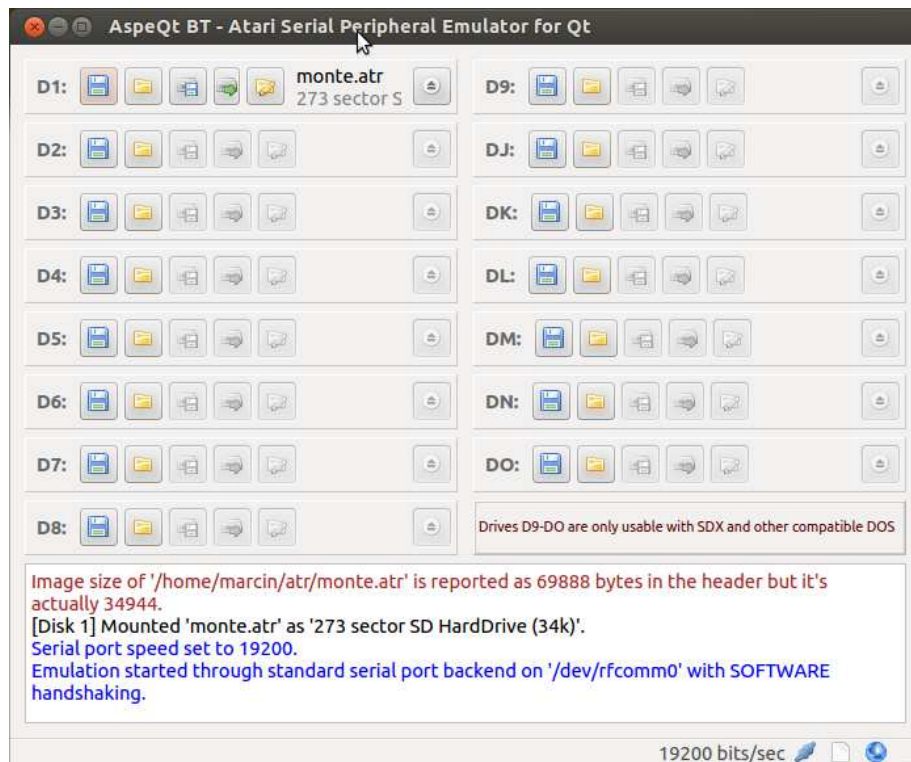
```
$ sudo apt-get install zlib1g-dev
```

```
$ qmake
```

```
$ make
```

And now the tool can be started:

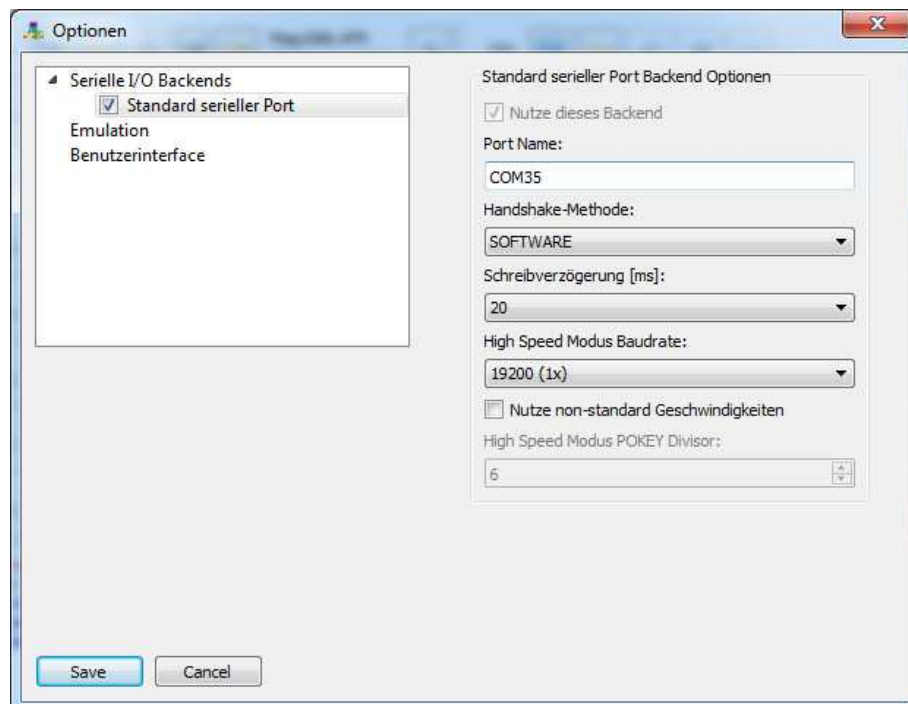
```
$ ./aspeqt
```

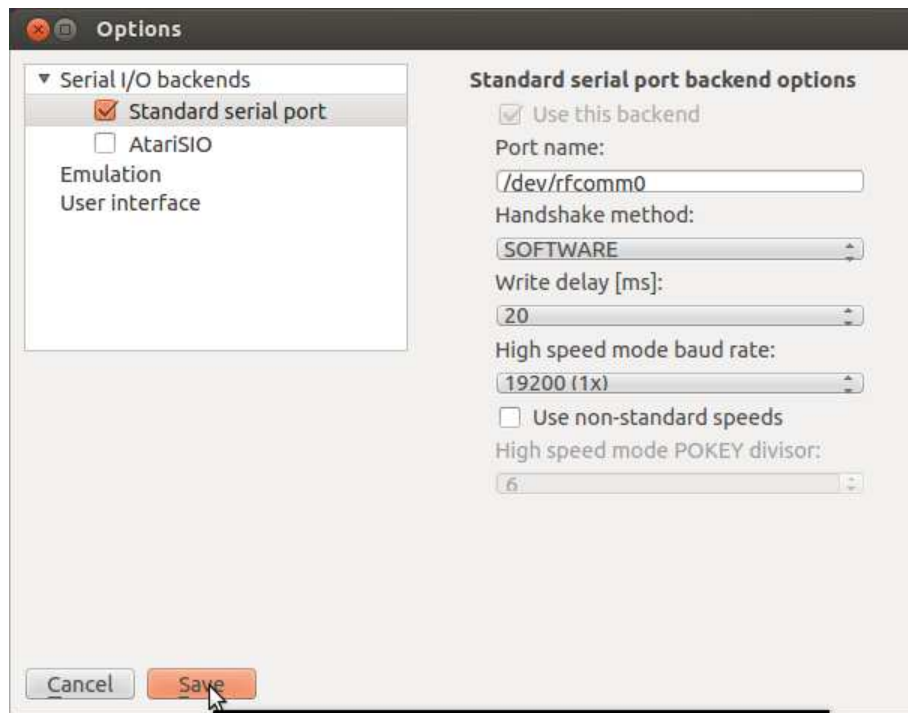


There is a new “SOFTWARE” handshaking method and a new “Write Delay [ms]” parameter introduced. It's default value is 10 milliseconds.

You can increase it if you observe communication problems with the SIO2BT and you can set it to 0 if you want to use this version of AspeQt with serial cables supporting only Rx/D and Tx/D.

Selecting SOFTWARE handshaking tells AspeQt to ignore the Command Line status. It processes incoming data from Atari (Data Out) and tries to detect valid command frames.





Emulating SIO devices with SIO2BT Android App

The Android App (SIO2BT) can be downloaded from Google Play:

<https://play.google.com/store/apps/details?id=org.atari.montezuma.sio2bt>

The App emulates up to 4 floppy disks.

You can select disk images (*.atr) and executable files (*.xex, *.com, *.exe).

The SIO2BT app uses the XEX Loader from the SDRIVE project.

The XEX files are visible to DOS as disks with only one entry (the XEX file itself).

MyDos / SPARTA DOS X users can mount just downloaded XEX files and copy them onto the harddrive (for example: KMK IDE2 / Ultimate+SIDE2). This way the flexibility of the Android app can be combined with high performance of the PBI based solutions. DOS 2.0 / 2.5 can handle up to 80KB big XEX files. QMEG, MyDos and SPARTA DOS X can handle bigger XEX files as well.

A "long touch" on an executable allows you to choose the xex loader address (default value is \$700). Write protection mode (R/RW) for a disk image can be set with a "long touch", too.

The emulated disks (unless write protected) can be modified (SIO commands: format, write sector, etc. are supported).

In case of communication problems, please increase the delay value in the settings menu.

If the Bluetooth connection is established, it stays alive even if the display is switched off or the user pressed the power button. This behavior is intentional, but it can drain the device battery.

Note: SIO2BT Android App works only with bundled hardware (please contact montezuma@abbuc.de for details)

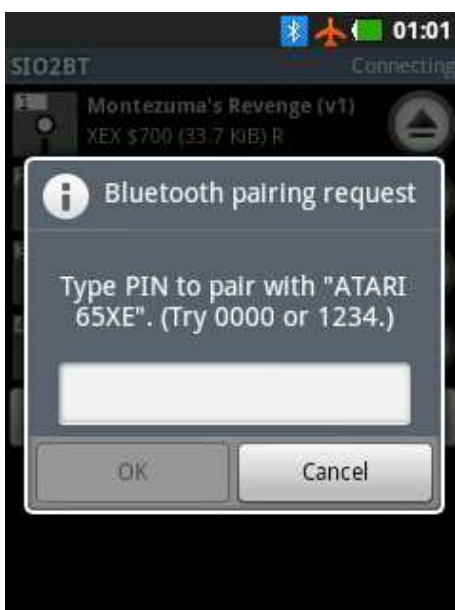
The following screenshots demonstrate briefly how to use the app.



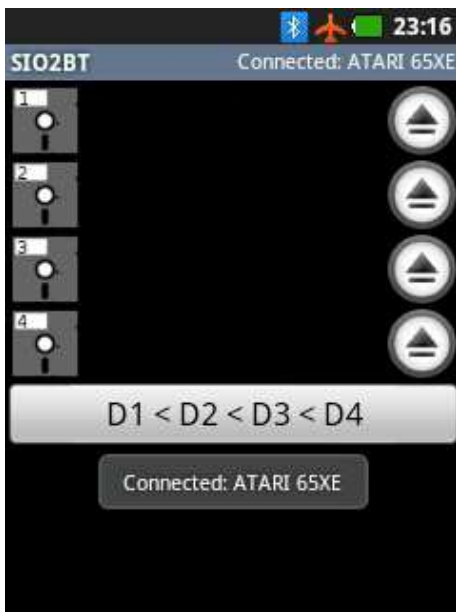
If you have not enabled Bluetooth on your Android device, before starting SIO2BT App, you will be asked to do so.



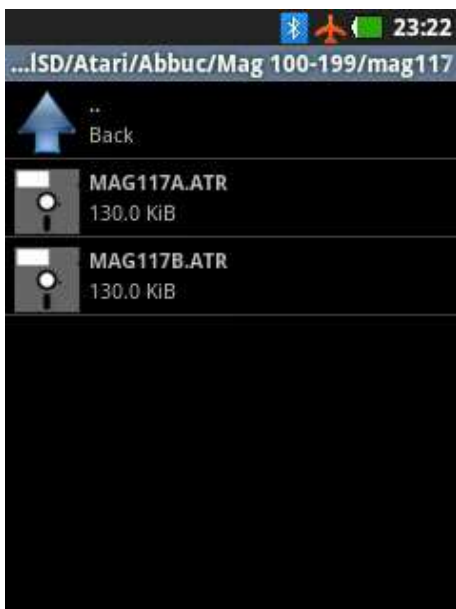
The App presents a list of paired SIO2BT devices. You may select one of those devices or touch „Scan for devices“ to search for and pair another SIO2BT device.



In case of a new SIO2BT device you will be prompted to enter the PIN code.



The connection status is shown in the right upper corner. Touching a disk icon or the place between the disk and eject icons starts the file selection activity.

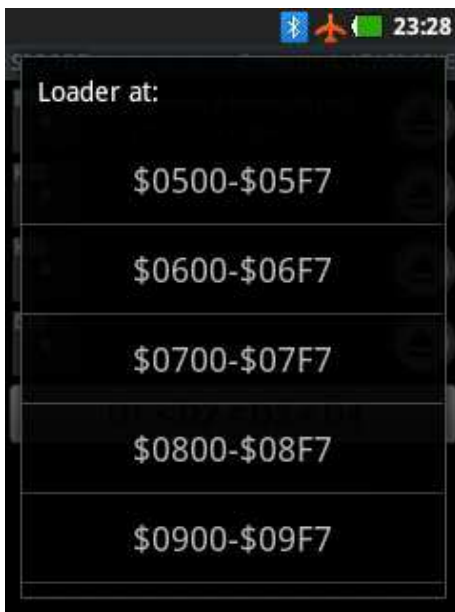


You may navigate through the file system by touching the „back“ icon or by touching directory names to any directory containing Atari files.

Touching (or long-touching) an ATR or XEX selects the file for mounting and saves the current directory information.



Long-touching an ATR file allows to select the access mode (R - read only, RW - read/write).



Long-touching a XEX file allows to select the loader address.



The SIO2BT is ready to work.
The button at the bottom of the screen can be used to quickly swap the disks (useful for multidisk games).



The App Menu has 4 items:

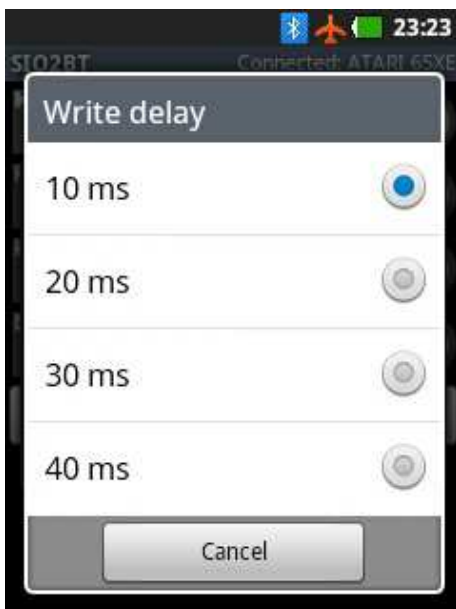
- “Connect” / “Disconnect” (depending on the current connection state) allows you to manage Bluetooth connections to SIO2BT devices
- “Create new disk” creates a DOS 2.5 formatted single density ATR file in the current directory
- “Settings” opens settings menu
- “Exit” closes current Bluetooth connection and exits the app



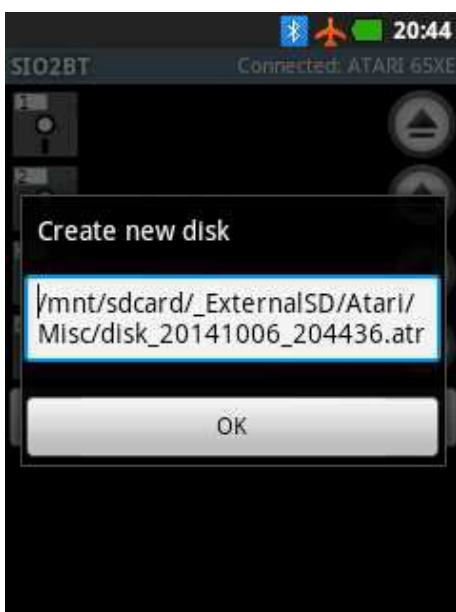
Settings menu has 2 items:

- “Write Delay” - delay required for stable communication
- “SIO LED” - SIO visualization

Both items may influence communication stability. The first one can be used to improve it. The second one simply causes additional CPU load.



The bigger the delay, the more reliable is the SIO communication.



The proposed name for a new ATR file is unique (contains the creation date and time).

Unless you change the path, it will be created in the current directory (directory of the recently mounted file).

The new ATRs are SD DOS 2.5 disks, but they can be formatted by any DOS to others formats.

Sparta DOS X support for SIO2BT

Sparta DOS X uses it's own SIO procedures per default.

The user has to tell Sparta DOS X to use the Atari OS procedures (patched for SIO2BT) in order to exchange data via Bluetooth.

To achieve the goal, please create the CONFIG.SYS file on the Sparta DOS X bootable partition.

The command below copies the default CONFIG.SYS file to the partition D1.

```
TYPE CAR:CONFIG.SYS >>D1:CONFIG.SYS
```

Edit the file: ED CONFIG.SYS

and append /A after the line: DEVICE SIO (it should like: DEVICE SIO /A).

Save the file and restart the Atari.

Note:

If you use Ultimate1MB with SIDE2, you must configure them in the following way:

- **SIDE2 switch should be in the XEX loader position**

- **Ultimate1MB settings:**

SPARTADOS X: ENABLED

SIDE HARDWARE: ON, WITH BUTTON

References

Bluetooth Essentials for Programmers, Albert S. Huang, ISBN-13: 978-0521703758

<http://en.wikipedia.org/wiki/Bluetooth>

<https://www.bluetooth.org>

http://wiki.pinguino.cc/index.php/SPP_Bluetooth_Modules

<http://atariage.com/forums/topic/201133-os-source-code-all-revisions/> (OS source code)

<http://www.atari8.co.uk/uflash/index.html> (uFlash)

<http://sourceforge.net/projects/aspeqt/> (AspeQt)

<http://drac030.krap.pl/en-inne-pliki.php> (sio2bsd)

<http://raster.infos.cz/atari/hw/sdrive/sdriveen.htm> (SDRIVE)

<http://www.lotharek.pl/product.php?pid=99> (Ultimate + SIDE 2)

<http://sdx.atari8.info> (Sparta DOS X)

<https://play.google.com/store/apps/details?id=org.atari.montezuma.sio2bt> (SIO2BT App)

<http://xxl.atari.pl/?p=1076> (XBIOS)